VIRTUALISATION ET OPENSTACK

ARNAUD MORIN

OBJECTIFS

- Virtualisation
- Cloud
- OpenStack

VIRTUALISATION

OBJECTIFS

- Principes et intérêts
- Vocabulaire
- Vue d'ensemble des solutions dispo

DÉFINITION

DÉFINITION

Selon wikipedia :
La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d’exploitation comme un simple logiciel, sur un ou plusieurs ordinateurs (serveurs), au lieu de ne pouvoir en installer qu’un seul par machine.

PRINCIPES DE POPEK ET GOLDBERG

En 1974, Popek et Goldberg sont deux chercheurs qui ont introduits des conditions pour qu’un système supporte la virtualisation :

PRINCIPES DE POPEK ET GOLDBERG

En 1974, Popek et Goldberg sont deux chercheurs qui ont introduits des conditions pour qu’un système supporte la virtualisation :

PRINCIPES DE POPEK ET GOLDBERG

En 1974, Popek et Goldberg sont deux chercheurs qui ont introduits des conditions pour qu’un système supporte la virtualisation :

PRINCIPES DE POPEK ET GOLDBERG

En 1974, Popek et Goldberg sont deux chercheurs qui ont introduits des conditions pour qu’un système supporte la virtualisation :

HISTORIQUE

- 1946 - Premiers ordinateurs "Turing-complet" (ex : ENIAC)
- 1958 - Ordinateurs multitaches (Gamma 60 de Bull) : faire tourner plusieurs programmes en même temps, concept proche de la virtualisation.
- 1972 - IBM Mainframe Virtual Machine Facility/370 : premier système de "full virtualisation" !

HISTORIQUE

- 1990 - Emulation de processeurs x86, mac sur Amiga (pionnier du genre)
- 1999 - VMWare Worsktation, puis Qemu, KVM, bochs, Xen, etc.
- 2004 - Intel VT-x Les VM ont directement accès au CPU. Les hyperviseurs ne font plus d'émulation mais controllent qui a accès au CPU

LES PRINCIPES DE POPEK ET GOLDBERG SONT RESPECTÉS DEPUIS 2004 SEULEMENT !

INTÉRÊTS DE LA VIRTUALISATION

INTÉRÊTS DE LA VIRTUALISATION

- Sécurité : isolation / cloisonement

INTÉRÊTS DE LA VIRTUALISATION

- Sécurité : isolation / cloisonement
- Coût : mutualisation / allocation temporelle

INTÉRÊTS DE LA VIRTUALISATION

- Sécurité : isolation / cloisonement
- Coût : mutualisation / allocation temporelle
- Criticité : sauvegarde / clonagee / migration

INTÉRÊTS DE LA VIRTUALISATION

- Sécurité : isolation / cloisonement
- Coût : mutualisation / allocation temporelle
- Criticité : sauvegarde / clonagee / migration
- Performance : allocation dynamique de resources

COMPRENDRE LA VIRTUALISATION

CPU X86: LES ANNEAUX DE PROTECTIONS

LES ANNEAUX DE PROTECTIONS

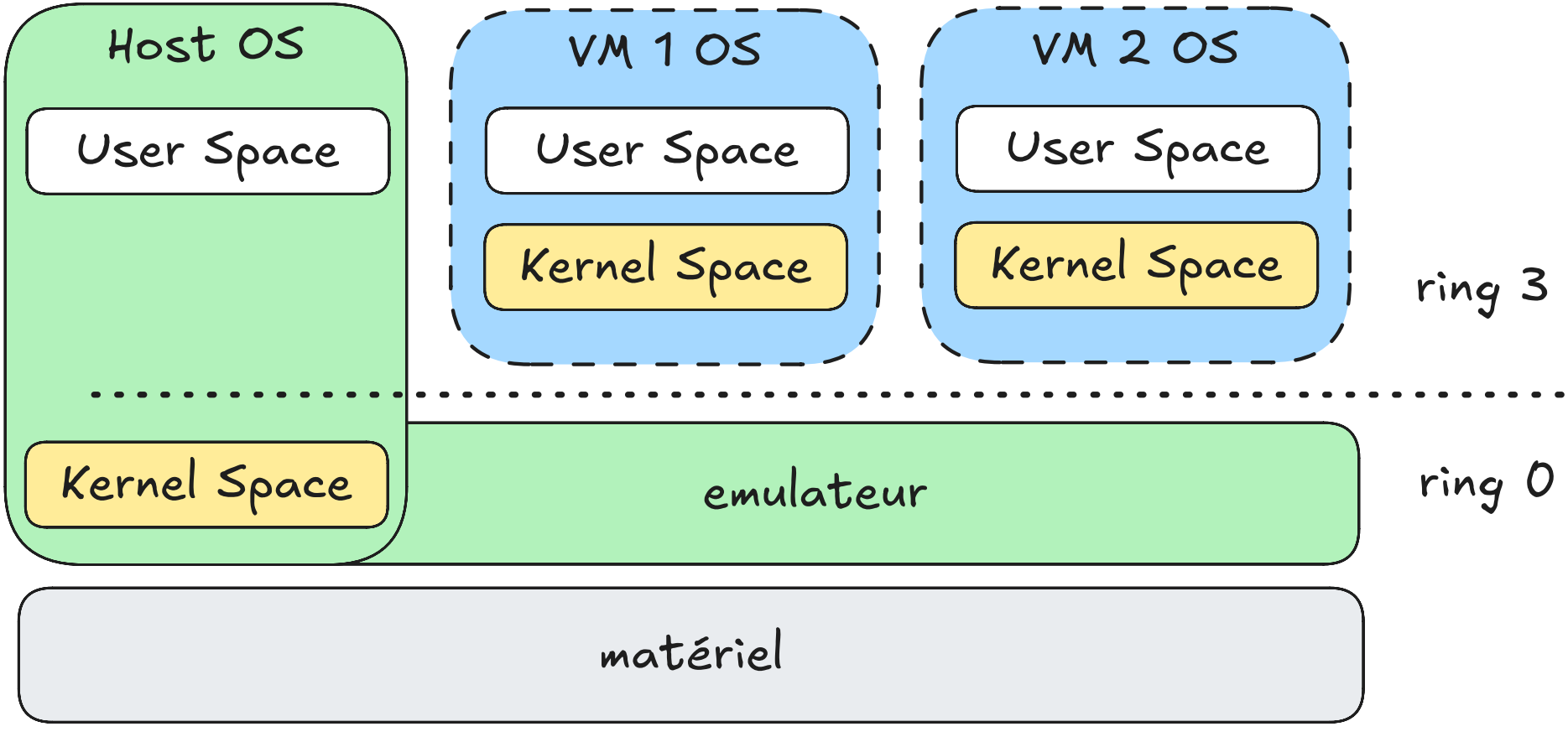
- Le "noyau" linux (ou windows) tourne dans le ring 0
- Les "logiciels" utilisateurs tournent dans le ring 3
- On passe d'un ring a l'autre en faisant un "sys call" (SYSENTER)

RAPPEL

La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d’exploitation comme un simple logiciel

LES ANNEAUX DE PROTECTIONS

- L'hyperviseur tourne dans le ring 0
- Les machines virtuelles dans le ring 3 (simples logiciels)


Host OS
User Space
Kernel Space
VM 1 OS
User Space
Kernel Space
VM 2 OS
User Space
Kernel Space
ring 3
ring 0
emulateur
matériel

La VM tourne comme un simple logiciel (ring 3).

Principe de Popek et Goldberg :

- équivalence : OK
- efficacité : KO
- contrôle : OK

CE N'EST PAS DE LA VIRTUALISATION !

On parle dans ce cas d'émulation

EMULATION

Pros:

- Bonne isolation entre les OS invités
- Cohabitation d’architecture CPU et OS hétérogènes

Cons:

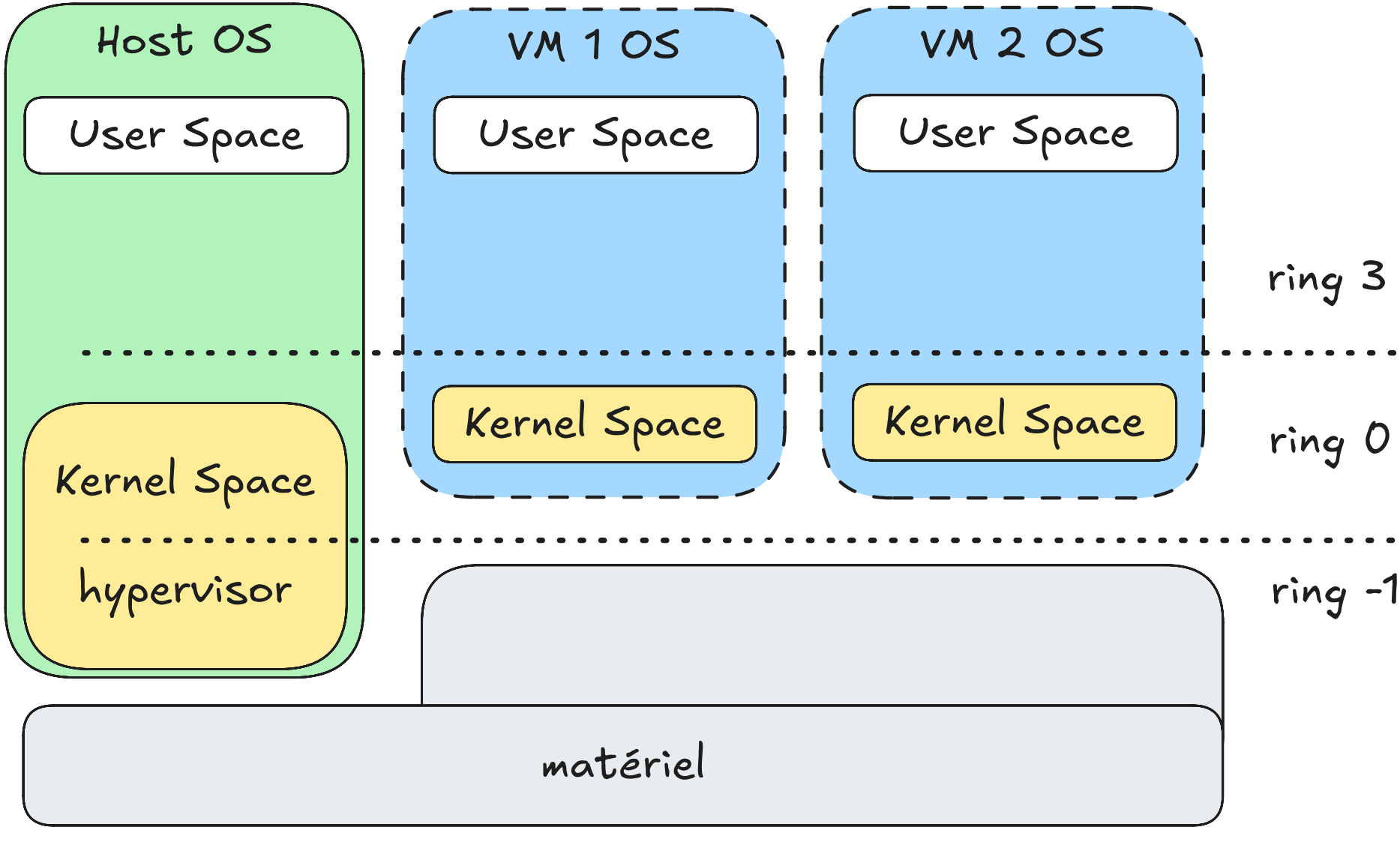
- Pas très performant : l'émulation provoque beaucoup d'overhead

RING -1

2004 : Intel et AMD ont ajouté à leurs processeurs des instructions CPU supplémentaires pour aider à la virtualisation :

- Intel VT-x
- AMD-V

Ces nouvelles instructions sont regroupées dans le Ring -1"


Host OS
User Space
Kernel Space
hypervisor
VM 1 OS
User Space
Kernel Space
VM 2 OS
User Space
Kernel Space
ring 3
ring 0
ring -1
matériel

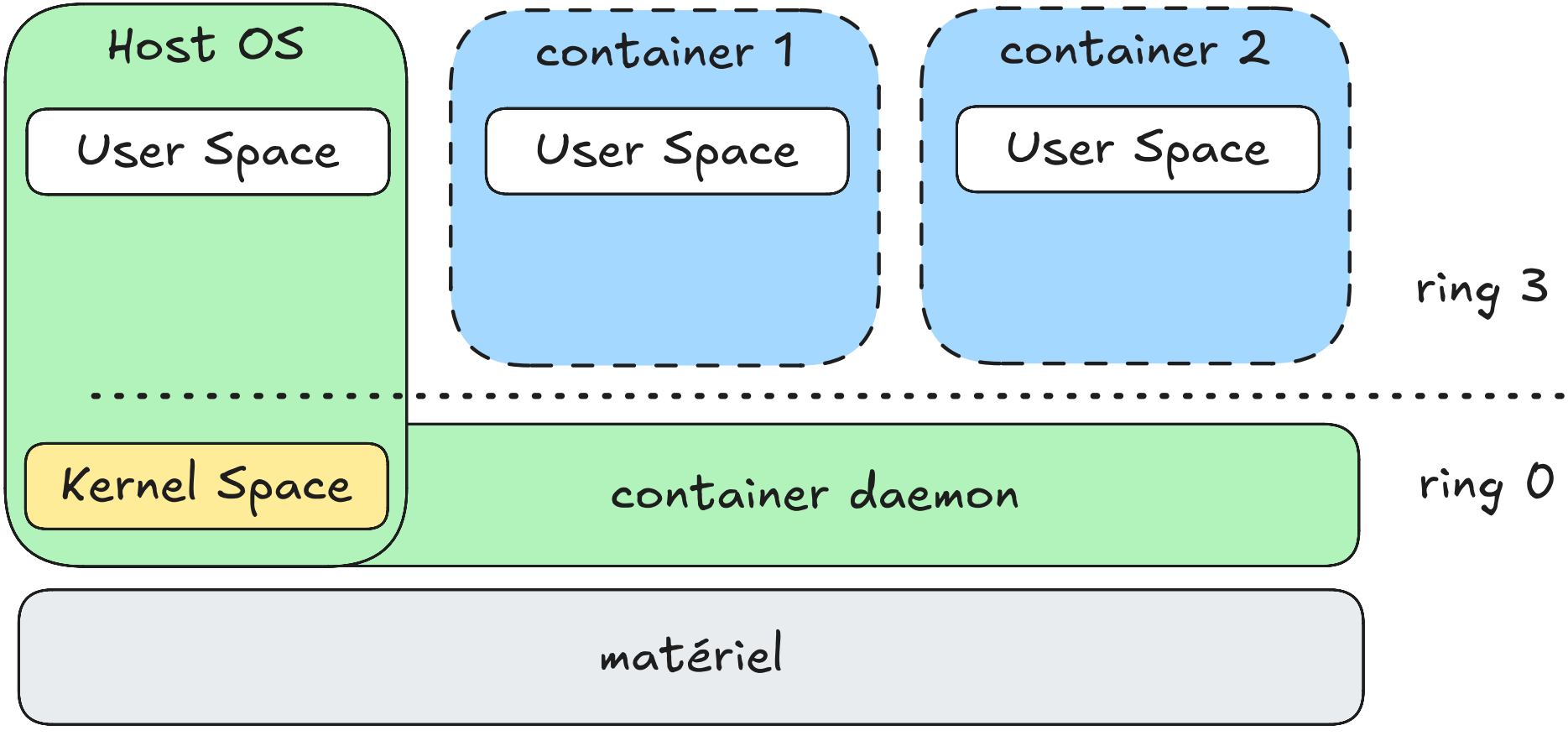
VIRTUALISATION

La VM à accès au ring 0.

Principe de Popek et Goldberg :

- équivalence : OK
- efficacité : OK
- contrôle : OK

CONTAINERS


Host OS
User Space
container 1
User Space
container 2
User Space
ring 3
Kernel Space
container daemon
ring 0
matériel

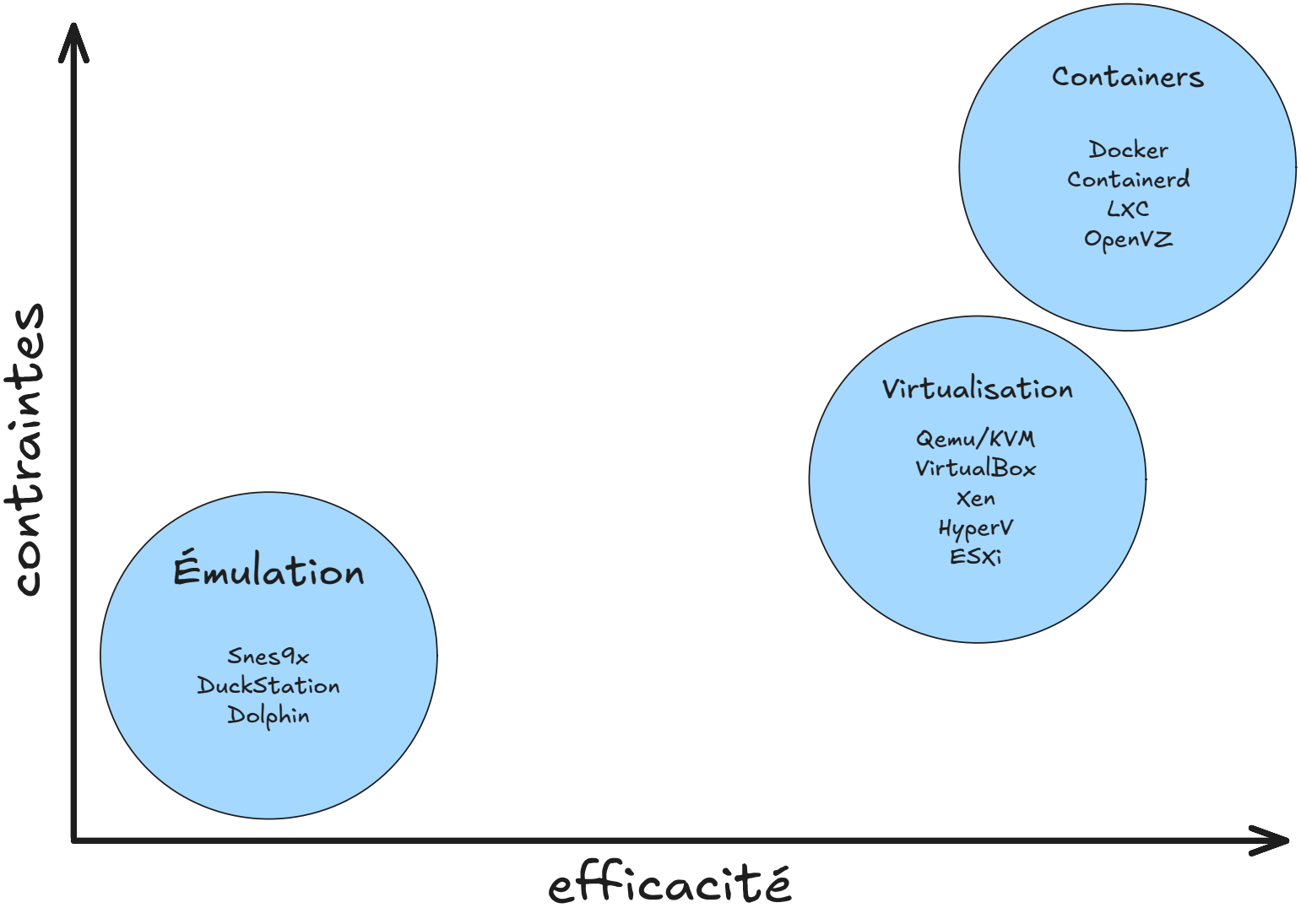
CONTAINERS

Ce n'est pas de la virtualisation.

Principe de Popek et Goldberg :

- équivalence : KO
- efficacité : OK
- contrôle : OK

LES MODELES DE VIRTU



CLOUD

CLOUD

- Principes et intérêts
- Vocabulaire et philosohie
- Vue d'ensemble des solutions dispo

LE CLOUD C'EST LARGE !

- Calcul / Virtualisation
- Stockage
- Abstraction du matériel
- Service et facturation à la demande
- Accès par des API REST
- Flexibilité, élasticité

WHAT YOU WANT AS A SERVICE

Principalement :

- IaaS : Infrastructure as a Service
- PaaS : Platform as a Service
- SaaS : Software as a Service

Mais aussi :

- Database as a Service
- Network as a Service
- Firewall as a Service
- Load Balancer as a Service
- DNS as a Service
- ...

TYPE DE CLOUD

- Cloud Public : fourni par un hébergeur à des clients (OVHcloud, AWS, GCP, Azure, Dropbox, etc.)
- Cloud Privé : interne a une entreprise
- Cloud Hybride : utilisation de ressources public au sein d'un cloud privé

POURQUOI FAIRE DU CLOUD

Coté business :

- Baisse des coûts par mutualisation
- Utilisation uniquement des ressources nécessaires

POURQUOI FAIRE DU CLOUD

Coté tech :

- Accès par des API / automatisation / agilité
- Reproductibilité
- Architectures résiliantes et scalables
- Abstraction des couches basses

CLOUD PROVIDERS

VIRTUALISATION DANS LE CLOUD

Le cloud computing repose souvent sur la virtualisation.

Le plus souvent en se basant sur l'hyperviseur Open Source QEMU/KVM

STOCKAGE DANS LE CLOUD

On distingue deux types de stockage : le block storage et l'object storage

BLOCK STORAGE

Stockage bas niveau

- Utilisé pour créer des disques virtuels (raw devices e.g. /dev/vdb)
- Compatible avec n'importe quel systeme de fichier
- Latence faible, idéal pour optimiser les I/O
- E.G. Amazon EBS, OpenStack Cinder

OBJECT STORAGE

Stockage haut niveau

- Accessible via une API HTTP
- Utilisé pour les fichiers non structurés (images, videos, logs, etc.)
- Optimisé pour la scalabilité et la durabilité
- E.G. Amazon S3, OpenStack Swift

LEXIQUE DU CLOUD : LES MOTS CLÉS À MAÎTRISER

API REST

Interface pour interagir avec les services cloud via des requêtes HTTP (GET, POST, etc.).

Comme un "menu de restaurant" : tu commandes (GET /instances) et tu reçois une réponse (liste des instances).

API DE METADATA/USERDATA

API interne permettant à une instance cloud d'accéder dynamiquement à des informations de configuration ou des scripts au démarrage, généralement accessible à l'URL 169.254.169.254.

Comme une "boîte aux lettres" accessible depuis une instance : les metadata sont les courriers administratifs alors que les userdata sont des colis personnels.

CLOUD-INIT

Outil pour initialiser une instance (ex: créer un utilisateur, installer des paquets). Il récupère les informations depuis l'API de metadata/userdata.

Comme un "assistant de première configuration" : il prépare la machine à ton arrivée.

IMAGE

Modèle préconfiguré d’un système d’exploitation (OS) ou d’une application, utilisé pour démarrer des instances.

Comme une "clé USB bootable" : permet de démarrer une machine avec un OS déjà installé (ex: Debian, Ubuntu, CentOS).

INSTANCE

Machine virtuelle (VM) ou conteneur exécutant un OS ou une application, créée à partir d’une image.

Une "boîte" avec CPU, RAM, disque, port.

FLAVOR

Modèles prédéfinis de ressources (CPU, RAM, disque) pour une instance.

Comme des "menus" au restaurant : petit (1 CPU, 2 Go RAM), moyen (4 CPU, 8 Go RAM), etc. Plus c'est gros, plus c'est cher !

VOLUME

Disques durs virtuels persistants, attachables/détachables à des instances. Se repose sur une technologie de block storage.

Comme un "disque dur externe" : on peut l'attacher sur n'importe quelle instance.

FLOATING IP

Adresses IP publique flottante, attachable à une instance pour un accès internet.

Comme un "numéro de téléphone portable" : on peut le transférer d’un appareil à un autre.

SECURITY-GROUP

Règles de pare-feu pour contrôler le trafic réseau vers/entre les instances.

Comme un "videur de boîte de nuit" : il autorise ou bloque l’accès selon des règles.

KEYPAIR

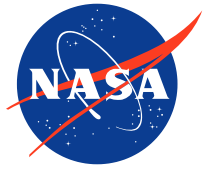
Clés SSH publiques/privées pour une connexion sécurisée aux instances.

Comme une "clé et une serrure" : la clé privée ouvre la porte (instance) verrouillée par la clé publique.

OPENSTACK

Suite logicielle Open Source pour construire un cloud.

HISTOIRE

Projet démarré en 2010 suite a la fusion d'un projet de la 

(cloud computing) et d'un projet de rackspace (cloud object storage)

Développé en  et distribué sous licence libre

Cycle de de 2 releases par an

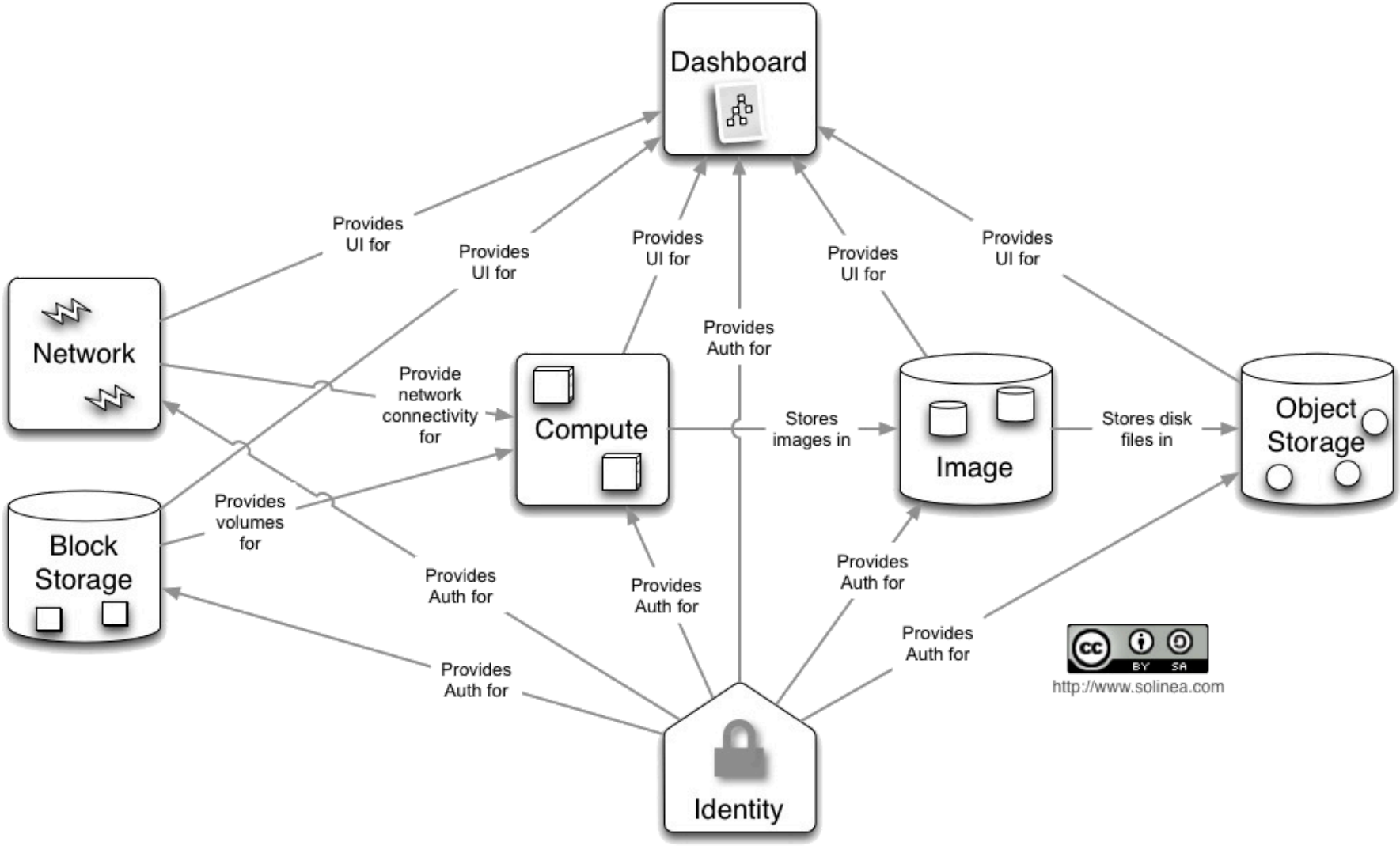
Les dernières releases :

- 2025.1 (epoxy)
- 2025.2 (flamingo)
- 2026.1 (gazpacho) - prévu pour avril

LES 4 OPENS

- Open Source
- Open Design
- Open Development
- Open Community

OVERVIEW


Dashboard
Network
Block Storage
Compute
Image
Object Storage
Identity
Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provide network connectivity for
Provides Auth for
Stores images in
Stores disk files in
Provides volumes for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
http://www.solinea.com

PRINCIPAUX PROJETS

- Compute : Nova 
- Network : Neutron 
- Image : Glance 
- Volume : Cinder 
- Authentication : Keystone 
- Object Storage : Swift 

MAIS AUSSI

OpenStack ne pourrait pas fonctionner sans d'autres briques Open Source, comme par ex. :

- Block storage : 

(pour cinder / volumes)
- Base de données : 

- Bus de messages : 

- etc.

INSTALLATION D'OPENSTACK

Installer OpenStack est relativement simple :

```
$ pip install nova
$ pip install neutron
# etc.
```

La complexité réside dans la configuration et l'imbrication des différents éléments entre eux.

Beaucoup d'options sont possibles et nécessite une forte expertise.

UTILISATION D'OPENSTACK

Pour discuter avec un Cloud OpenStack, le plus simple est d'installer et utiliser le client officiel :

```
$ pip install python-openstackclient
```

Ce client permet de faire des requêtes HTTP aux differentes API REST du cloud OpenStack.

FICHIER OPENRC

Le client `openstack` nécessite des paramètres pour se connecter au cloud OpenStack.

A minima, il faut :

- L'`URL` de l'API `keystone`
- Un `login`
- Un `mot de passe`
- Un identifiant `project`

EXEMPLE DE FICHIER OPENRC

```
$ cat openrc
export OS_AUTH_URL="https://auth.cloud.ovh.net/v3/"
export OS_TENANT_ID="0d899a6f76e74760a06919533ed0ec52"
export OS_USERNAME="user"
export OS_PASSWORD="password"
```

CATALOG

La premiere requête que nous pouvons faire avec le client `openstack` est une requête aupres du service `keystone` pour récupérer le `catalog` et ainsi découvrir les autres API disponibles dans notre cloud.

```
$ openstack catalog list
```

LISTE DES INSTANCES

Si le service `nova` (compute) est disponible sur le cloud, il devient possible de lister les instances :

```
$ openstack server list
```

LISTE DES IMAGES

Si le service `glance` (image) est disponible sur le cloud, il devient possible de lister les images :

```
$ openstack image list
```

LISTE DES RÉSEAUX

Si le service `neutron` (network) est disponible sur le cloud, il devient possible de lister les réseaux :

```
$ openstack network list
```

ETC.

Beaucoup d'autres requêtes peuvent être faite.

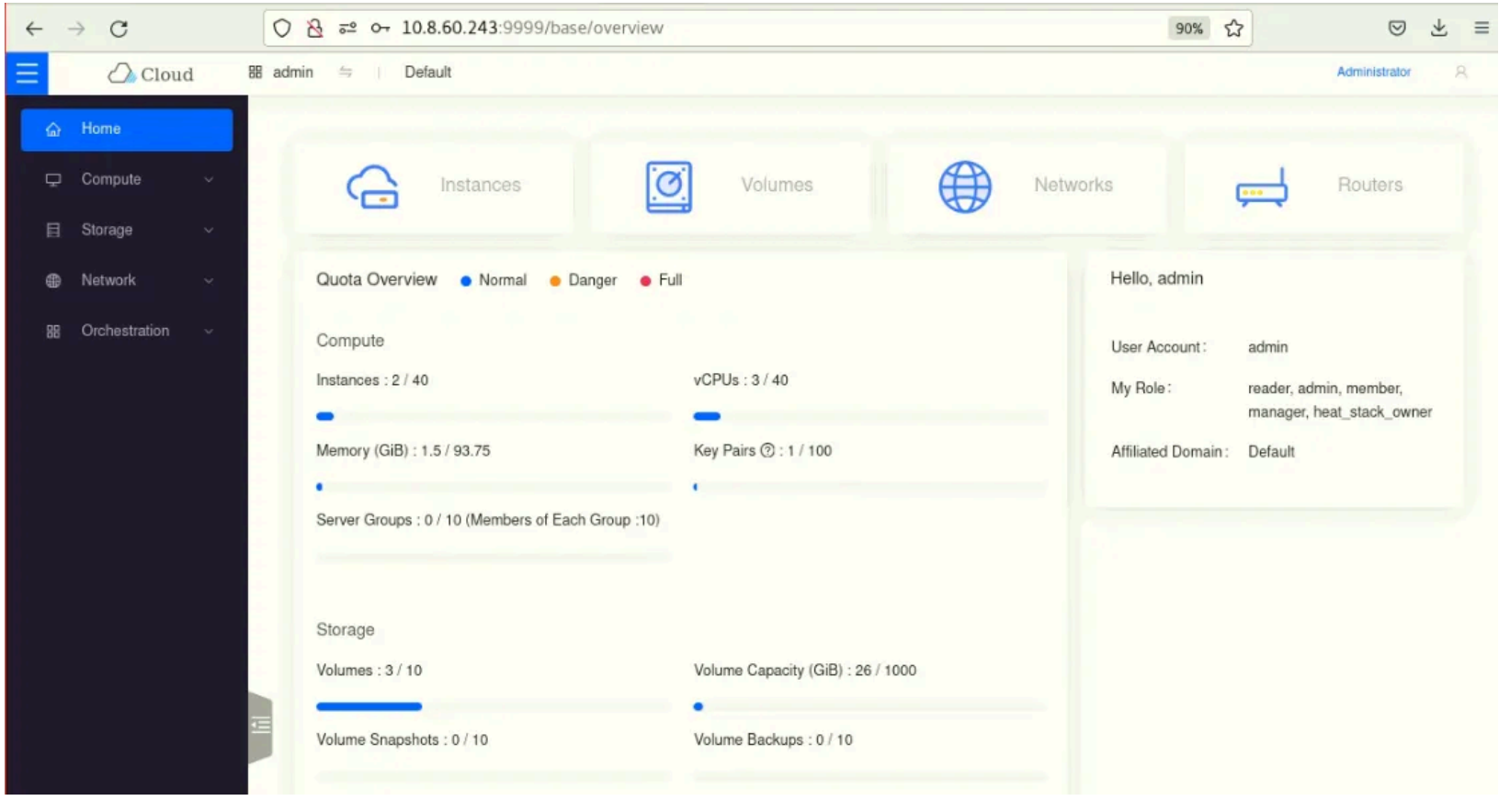
Comme par ex. booter une nouvelle instance

```
# Création d'une nouvelle instance
$ openstack server create \
  --flavor small \
  --image 'Debian 13' \
  --net public \
  mon-instance
```

etc.

INTERFACE WEB

Il est aussi possible de piloter OpenStack à partir d'une interface web.



...OU VIA DU CODE

```
# Exemple avec du code terraform
resource "openstack_compute_instance_v2" "mon_instance" {
  name        = "mon-instance"
  image_name  = "Debian 13"
  flavor_name = "small"
  network {
    name = "public"
  }
}
```


QUESTIONS ?